# Inleiding

Voor de studie HBO-ICT is er naar aanleiding van een projectopdracht een lasergame gemaakt wat uitgevoerd wordt op een Arduino Due. Dit lasergamesysteem maakt gebruik van een Real Time Operating System (RTOS) genaamd Arduino RTOS.

Naast de Due zijn er echter veel alternatieve systems on a chip (SOC's). Ontwikkeling van software producten is dan ook een moeilijke taak, want de vraag blijft altijd of de software ook nog steeds past op andere hardware. De RTOS is de basis waarop de rest van de software wordt gemaakt en niet ieder RTOS past op alle SOC's en microcontrollers.

In dit onderzoeksrapport wordt gekeken naar de verschillende alternatieve RTOS'en die verenigbaar zijn met het project. Niet alle RTOS'en zijn hier voor geschikt. Ten eerste moet een RTOS voldoen aan verschillende non functionele eisen, zoals ondersteuning van C en C++ en of het RTOS open source is of niet.

Daarnaast moet er duidelijk zijn welke functionaliteiten wel of niet aanwezig zijn in de RTOS'en die voldoen aan de non-functionele eisen. De kans is klein dat een RTOS specifiek alle functionaliteiten heeft, die het HU systeem ook biedt.

Om de verschillende eisen goed te kunnen onderzoeken heeft de opdrachtgever de volgende hoofdvraag opgesteld:

‘Met behulp van welk open source realtime operating system kunnen tasks en de concurrency mechanismen, pool, channel, flag(group), clock timer en mutex, zoals aangeboden door het Arduino RTOS, met zo weinig mogelijk overhead worden gerealiseerd?’

Om deze vraag zo gestructureerd mogelijk te beantwoorden zijn er de volgende deelvragen gegeven:

* Wat zijn de kenmerkende eigenschappen van tasks en de concurrency mechanismen van het Arduino RTOS?
* Welke open source RTOS-en zijn beschikbaar?
* Welk van de beschikbare RTOS-en biedt de meeste van de concurrency mechanismen van het Arduino RTOS aan zonder enige modificatie en biedt dezelfde functionaliteit om taken de realiseren
* Welke mechanismen van het Arduino RTOS worden niet ondersteund door de beschikbare RTOS-en?
* Hoe kunnen de mechanismen ban het Arduino RTOS die niet direct worden ondersteund door de beschikbare RTOS-en worden gerealiseerd met behulp van deze RTOS-en?

Op basis van deze deelvragen zal in het hoofdstuk onderzoeksmethoden het onderzoeksplan opgesteld. In het hoofdstuk resultaten zullen de deelvragen beantwoord worden. In het hoofdstuk conclusie wordt de hoofdvraag beantwoord.

Het onderzoek zal hoofdzakelijk bestaan uit een literatuurstudie die wordt uitgevoerd op het internet, op de documentatie van het Arduino RTOS en op de documentatie van de RTOS-en die in dit rapport worden onderzocht.

# Onderzoeksmethoden

Er zijn vijf verschillende eisen waar rekening mee gehouden moet worden bij het vinden van een geschikt RTOS. Deze eisen zijn geformuleerd in de vorm van de vijf deelvragen die in de inleiding zijn genoemd. In dit hoofdstuk wordt beschreven hoe deze vragen worden beantwoord.

## Arduino RTOS-functionaliteiten

Als er een alternatief wordt gebruikt voor het huidige RTOS moet deze ondersteuning bieden voor de beschikbare functionaliteiten. De vraag om deze eis te onderzoeken is ***"Wat zijn de kenmerkende eigenschappen van tasks en concurrency mechanismen van het Arduino RTOS".***

Er is een beeld van welke dit zijn. Voor dit onderzoek is er echter nog een beter overzicht nodig. Daarom zal de documentatie van het Arduino RTOS worden bestudeerd en op basis hiervan zal er een functionaliteiten lijst worden opgesteld.

## Kandidaat RTOS-en

Er zijn ongelooflijk veel verschillende RTOS-en, maar een duidelijk overzicht van welke beschikbaar zijn, is er niet. De deelvraag ***"Welke open-source RTOS-en zijn beschikbaar"*** wordt beantwoord door een lijst van RTOS-en op te stellen. Om de lijst van beschikbare systemen behapbaar te maken is er op basis van het huidige systeem een lijst van 4 non-functionele eisen opgesteld:

* Beschikbaar op het ARM-platform
* Ondersteuning voor C en C++
* Binnen het afgelopen jaar geüpdatet
* Beschikbare documentatie met beschrijving over geboden functionaliteiten

Door middel van een literatuuronderzoek zal er gezocht worden naar de verschillende RTOS-en die voldoen aan deze eisen.

## Compatibiliteit kandidaat RTOS-en

Het moment dat er een ingekorte lijst is gemaakt van alle RTOS-en die voldoen aan de non functionele eisen kunnen de functionaliteiten van deze systemen vergeleken worden. Om ***“Welk van de beschikbare RTOS-en biedt de meeste van de concurrency mechanismen van het Arduino RTOS aan zonder enige modificatie en biedt dezelfde functionaliteit om taken de realiseren”*** te beantwoorden bekijken we voor elk van de in deelvraag 2 gevonden RTOS-en of ze voldoen aan de lijst van functionaliteiten die is opgesteld voor deelvraag 1. De resultaten worden samengevoegd in een tabel zodat er een duidelijk overzicht van de compatibiliteit van de verschillende RTOS-en. Tabel 1 is een voorbeeld hiervan.

|  |  |  |
| --- | --- | --- |
|  | RTOS 1 | RTOS 2 |
| Functionaliteit 1 | V | X |
| Functionaliteit 2 | X | V |

Tabel 1: Voorbeeld structurering resultaten deelvraag 3

## Niet ondersteunde functionaliteiten

Het is belangrijk duidelijk te hebben welke eigenschappen niet worden gedekt door een systeem. Door middel van de gegevens die komen uit de resultaten van deelvraag 3 wordt ***“Welke mechanismen van het Arduino RTOS worden niet ondersteund door de beschikbare RTOS-en?”*** beantwoord. De tabel geeft een duidelijk overzicht van ondersteunde mechanismen en welke hiervan nog moeten worden geïmplementeerd.

## Implementatie niet ondersteunde functionaliteiten

Hoogstwaarschijnlijk is er niet een RTOS die alle benodigde functionaliteiten biedt. Sommige van de mechanismen zullen dus moeten worden geïmplementeerd met op maat gemaakte code. Voor elk van de mechanismen gevonden door middel van deelvraag 4 zal worden gekeken hoe deze moeten worden geprogrammeerd. Hiermee wordt de laatste deelvraag, “***Hoe kunnen de mechanismen van het Arduino RTOS die niet direct worden ondersteund door de beschikbare RTOS-en worden gerealiseerd met behulp van deze RTOS-en?”*** beantwoord. Daarnaast wordt er ook een inschatting gemaakt van de complexiteit van de implementatie.

De resultaten zullen in het hoofdstuk resultaten worden samengevat. Door deze data is het mogelijk een keuze voor een systeem te maken wat aan de criteria voldoet, zoveel mogelijk van de functionaliteiten heeft en hoe missende functionaliteiten binnen dit systeem kunnen worden geïmplementeerd. Deze keuze wordt in de conclusie gegeven en beargumenteerd.

# Resultaten

## Arduino RTOS-functionaliteiten

De Arduino RTOS-documentatie is doorzocht op de verschillende functionaliteiten. In eerste instantie is er gezocht op de in de hoofdvraag genoemde mechanismen. Na verder onderzoek zijn hier nog zeven extra functies bij gekomen. Deze functies liggen buiten het initiële bereik van het onderzoek, maar zullen wel meegenomen in de vergelijking. Dit wordt gedaan omdat sommige van de kandidaat RTOS-en niet over bepaalde functionaliteiten bezit, maar deze wel gebruikt zullen worden binnen een real time systeem.

|  |  |
| --- | --- |
| Functionaliteit | Beschrijving |
| Mutex |  |
| Eventflag |  |
| Mailbox |  |
| Channel |  |
| Timer |  |

Tabel 2: Overzicht Arduino RTOS-functionaliteiten

## Kandidaat RTOS-en

Het opstellen van de lijst van mogelijke RTOS-en is gedaan aan de hand van de lijst van RTOS-en op Wikipedia. Deze lijst heeft een beknopte samenvatting van kenmerken die gebruikt konden worden in het selectieproces. [RTOS lijst link vanuit wikipedia]

De lijst is eerst gefilterd op de beschikbaarheid op het ARM-platform. In de lijst op Wikipedia staat ook of een systeem actief wordt onderhouden. Alleen de actief onderhouden systemen zijn verder beoordeeld. De lijst had geen informatie over de ondersteuning van C en C++ of over beschikbare documentatie. Door middel van het onderzoeken van de overgebleven systemen is er een lijst opgesteld van potentiele RTOS-en. Tabel 2 is een overzicht van deze 10 RTOS-en.

|  |  |
| --- | --- |
| RTOS | Beschrijving |
| EcosPro |  |
| uKOS |  |
| cocoOS |  |
| Distortos |  |
| FreeRTOS |  |
| FunkOS |  |
| scmRTOS |  |
| NuttX |  |
| StratifyOS |  |
| mbed OS |  |

Tabel 3: Overzicht van kandidaat RTOS-en

## Compatibiliteit kandidaat RTOS-en

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **RTOS naam** | **mutex** | **event flag** | **channel** | **timer** | **pool** |
| EcosPro | V | V | X | V | X |
| uKOS | V | V | V | V | X |
| Coco OS | V | V | V | V | V |
| Distort OS | V | V | X | V | X |
| FreeRTOS | V | V | X | V | X |
| Funk OS | V | V | X | V | X |
| scmRTOS | V | V | V | V | V |
| NuttX | V | V | V | V | V |
| Stratify OS | V | V | X | V | X |
| MbedOS | V | V | X | V | V |

Tabel 4: Vergelijking kandidaat RTOS-en met Beschikbare functionaliteiten

Van de tien onderzochte systemen zijn er drie systemen die alle functies hebben, Nuttx, scmRT OS en Coco OS.

## Niet ondersteunde functionaliteiten

Aan de hand van de resultaten in tabel vier, zijn er twee functionaliteiten gevonden die missen, de channel en de pool In tabel 5 is te zien welke functionaliteiten dit zijn en welke systemen ze missen.

|  |  |
| --- | --- |
| Functionaliteit | Niet ondersteunende RTOS-en |
| Channel | EcosPro  FreeRTOS  Stratify OS  Distort OS  Funk OS  MbedOS |
| Pool | EcosPro  Distort OS  Funk OS  MbedOS  uKOS  FreeRTOS  Stratify OS |

Tabel 5: Overzicht missende functionaliteiten

## Implementatie niet ondersteunde functionaliteiten

De missende functionaliteiten kunnen geïmplementeerd worden door andere functionaliteiten te gebruiken. Voor elk van de functionaliteiten is in tabel 6 te zien hoe deze geïmplementeerd kunnen worden.

|  |  |
| --- | --- |
| Functionaliteit | Implementatie |
| Channel | Maak een message queue aan  Zit in de 1e waarde de klasse naam  De andere klasse zoekt naar de queue waarvan de 1e waarde de naam van de klasse is waarmee het communiceert |
| Pool | Maak een stack aan  Values worden op de stack gezet  Opvragen:  Forloop door de stack tot het object  Haal object uit de stack  Krimp de stack in |

# Conclusie

Er zijn op dit moment [heel veel] RTOS-en. Veel hiervan voldoen niet aan de criteria van het Devices project. Een duidelijk overzicht van systemen is er niet en documentatie is regelmatig afwezig. Na [heel veel] bekeken te hebben via de Wikipedia lijst zijn er twee systemen uitgekomen die voldoen aan alle benodigde functionaliteiten, NuttX en scmRT OS. Een keuze voor een RTOS hoeft echter niet te vallen op een van deze twee. Het is mogelijk om missende functionaliteiten eenvoudig te implementeren in een van de andere acht systemen. De conclusie van dit rapport is echter wel dat het antwoord op de hoofdvraag, ***‘Met behulp van welk open source realtime operating system kunnen tasks en de concurrency mechanismen, pool, channel, flag(group), clock timer en mutex, zoals aangeboden door het Arduino RTOS, met zo weinig mogelijk overhead worden gerealiseerd?’*** NuttX is.

# Discussie

In dit rapport is voornamelijk gekeken naar functionaliteiten. Een van de vervolgstappen die bij dit marktonderzoek gedaan kan worden is het uitbreiden van zowel de non-functionele eisen als benodigde functionaliteiten.

Met het oog op non-functionele eisen is performance heel erg significant. NuttX is een uitgebreid systeem en de vraag is of de omvang van het pakket niet een negatieve invloed heeft op de werking van het real time systeem.

Qua functionele eisen is het interessant om te kijken hoe bepaalde concurrency mechanismen worden geïmplementeerd. Het is mogelijk dat bepaalde implementatie niet ideaal aansluiten op de door deze projectgroep geschreven code, of op de HWLIB van de hogeschool Utrecht. Het kan dat een systeem met meerdere missende functionaliteiten beter aansluit door andere functionaliteiten.

Daarnaast is het de vraag of de functionaliteiten die het Arduino RTOS biedt, optimaal zijn voor het opzetten van een lasergamesysteem. Een van de functionaliteiten van het NuttX is een spinlock. Binnen het te bouwen systeem is een spinlock een interessante toevoeging. Een vervolgonderzoek is mogelijk om te kijken wat de functionaliteiten zijn die nodig zijn voor een lasergame systeem.